Lab 4 Report

1st Skyler Cook
Electrical and Computer Engineering
Utah State University
Logan, USA
A02304185@aggies.usu.edu

2nd Jaron Seedal
Electrical and Computer Engineering
Utah State University
Logan, USA
a02358703@aggies.usu.edu

3rd Dr. Jonathan Philips
Electrical and Computer Engineering
Utah State University
Logan, USA
jonathan.phillips@usu.edu

***Abstract*—This document is a written report of the laboratory work performed to implement a simulated annealing algorithm for the FPGA placement problem. The program ingests a text file describing a grid, nodes, and their interconnections, and produces an output file listing the node placements and edge lengths. The goal of the lab was to minimize the total wire length by optimizing node positions on the grid.**

I. Introduction

This project implements simulated annealing as a heuristic optimization technique for the FPGA placement problem. In the placement problem, nodes of a circuit must be assigned to locations on a grid such that the sum of the interconnection lengths between nodes is minimized. Because placement is NP-hard, exact methods are impractical for large graphs, and heuristic methods like simulated annealing are widely used.

The program was written in C to follow a structured optimization loop to place nodes on a grid to minimize total interconnection length. At each iteration, two nodes swap positions, the total cost is recalculated, and the move is accepted or rejected based on the annealing schedule. The final result is a node placement that produces a relatively short total interconnection length while avoiding local minima.

II. Objectives

The objectives of the project, as defined by the instructor, were:

1) Write a simulated annealing program in C or C++ that ingests a text input file describing a grid size, number of nodes, and edges between nodes.
2) Assign nodes to grid coordinates, ensuring no overlap.
3) Minimize the total distance of all edges using simulated annealing with an appropriate cooling schedule.
4) Write the resulting node placements and corresponding edge lengths to an output text file.
5) Perform a statistical analysis of the relationship between cooling rate, execution time, and solution quality.

III. Procedure

The students began by reviewing the example simulated annealing code provided by the professor for the traveling salesman problem. The overall structure of this code was maintained, with modifications made to adapt it to the placement problem.

The first step was to create a hard-coded version of the program to confirm understanding of the algorithm and to verify that the simulated annealing framework functioned correctly. In this version, the grid size, number of nodes, and edges were directly written into the source code. The program initialized the node placement by assigning each node to a unique grid coordinate, then iteratively swapped node positions, evaluated the resulting total edge length, and accepted or rejected moves based on the annealing temperature.

After confirming that the hard-coded version worked as expected, the program was extended to support file input and output. The input file defined the grid dimensions, number of nodes, and a list of edges between nodes, while the output file listed the placement of each node and the length of each edge. The program was modified to parse this input format, initialize node placements on the grid, and then write the final results in the required format.

The simulated annealing loop was left unchanged from the initial prototype: a solution was copied, altered by swapping two nodes, evaluated by summing the Manhattan distances of all edges, and either accepted or rejected based on the temperature schedule. The program continued until the temperature cooled below the stop threshold.

Once the program was functional with file input and output, tests were performed with the example input provided in the lab requirements and with additional larger examples to study how cooling rate and runtime affected solution quality.

IV. Results

The program successfully read input files, placed nodes on the grid, and wrote the final placements and edge lengths to an output file. The simulated annealing process consistently reduced the overall edge lengths compared to the initial placement. Tests with different cooling schedules showed that slower cooling produced better minimized results but required longer runtimes, while faster cooling finished quickly but often settled on higher-cost solutions.

By testing, the programs results showed that a higher initial temperature did not reduce cost significantly, shown in Fig 1. However, a more intense cooling rate significantly reduced cost, shown in Fig 2. A lower stopping threshold reduced cost to a point, but did not help once it fell out of the cooling rate range, shown in Fig 3.

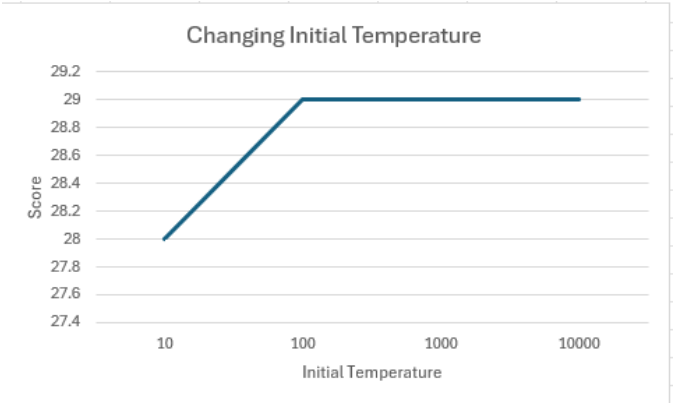

Fig. 1. Initial Temperature Chart

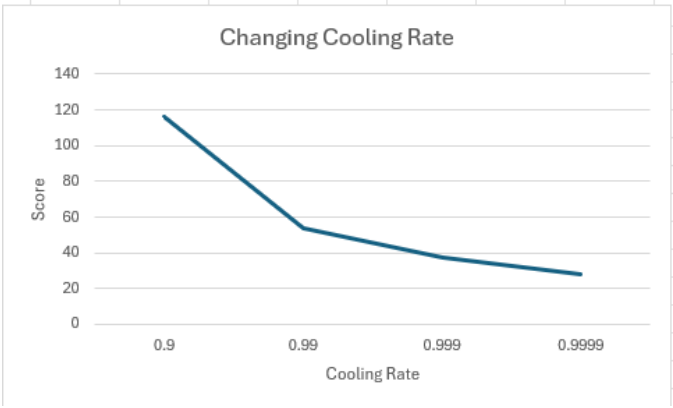

Fig. 2. Cooling Rate Chart

V. Discussion and Conclusions

The results demonstrate that the simulated annealing algorithm is effective at improving placement quality by significantly reducing total edge length relative to the initial configuration. As expected, the trade-off between cooling schedule and runtime was evident: slower cooling schedules achieved lower-cost solutions at the expense of increased execution time, while faster schedules favored speed over solution quality. Overall, the program confirmed the practical usefulness of simulated annealing as a heuristic method for solving the NP-hard placement problem.

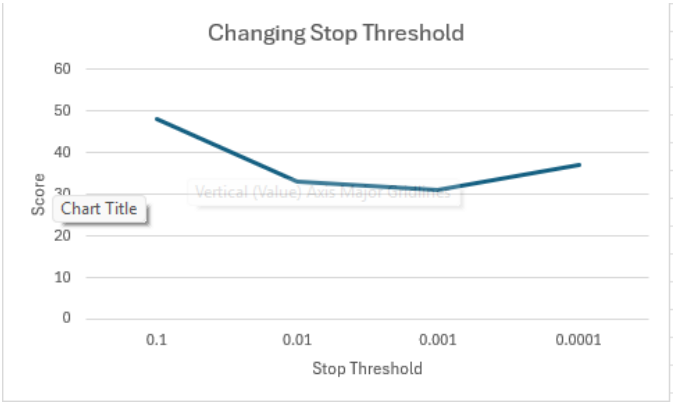

Fig. 3. Stop Threshold Chart